Programming – Working with Databases in C#

## Opgave 1

Lav et konsolprogram, der forbinder til din database med PET- og PET\_OWNER-tabellerne.  
  
Lav en menu med "Insert a Pet" som et valg. Når du vælges menupunktet, bliver du bedt om at indtaste oplysningerne for et nyt kæledyr. Kæledyret skal registreres i databasen ved hjælp af den Stored Procedure InsertPet, du oprettede i ExDB05, øvelse 1.  
  
Tilføj et menuvalg "Vis alle kæledyr", der bruger GetPets Stored Procedure, der er oprettet i ExDB05, øvelse 2 for at få alle kæledyroplysninger og udskriv dem på skærmen.  
  
Tilføj et menuvalg for at tilføje en kæledyrsejer på samme måde.  
  
Dernæst tilføj menuvalg "Find ejer af LastName" og "Find ejere via e-mail", som spørger om det nødvendige input og bruger derefter de Stored Procedurer GetOwnersByLastName og GetOwnerByEmail oprettet i ExDB05, øvelse 3 for at finde ejere - liste resultatet til skærmen .  
  
Endelig tilføj et valg "Ejers kæledyr", der bruger den sidste Stored procedure fra øvelse 3 for at udskrive resultatet af sammenhængen mellem PET og PET\_OWNER.  
  
Opdel din kode til metoder ved hjælp af SOLID-kriterier.

## Opgave 2

Når attributter i en tabel er erklæres som "*not null*", skal de indeholde en værdi.

Hvis den SP der bruges til at insætte en række prøver at oprette en række med en null værdi i en attributte der er erklæres som "not null" sinaleres der en fejl (exception).  
  
På samme måde kan en SP selv kaste en exception i tilfælde af kritiske fejl ved at bruge kommandoen THROW næsten på samme måde som i C#:  
  
For eksempel at kaste en exception, hvis en parameter indeholder NULL, kan du bruge dette eksempel:

IF @ParamName IS NULL THROW 51000, 'Værdien er NULL',1;  
  
Ændre SP InsertOwner, så der kastes en exception, hvis værdien for OwnerFirstName OG OwnerLastName allerede findes i tabellen.